**PICUR 2025**

**Integrantes**

|  |  |  |
| --- | --- | --- |
| **Nombre completo** | **Cédula** | **Correo electrónico** |
|  |  |  |
| Mateo Gutiérrez Sepulveda | 1038867589 | Mateo.gutierrez.7589@miremington.edu.co |
| Jorge David Mercado Serje | 1042247805 | Jorge.mercado.7805@miremington.edu.co |
| Breyner Yulian Becerra Taba | 1059697560 | Breyner.becerra.7560@miremigton.edu.co |

**Entrega 1 – Proyecto de Investigación Semana 1**

**Título del Proyecto:**

*Corrector inteligente de postura mediante dispositivo portátil con conectividad Wi-Fi e inteligencia artificial para la prevención de problemas musculoesqueléticos.*

**1. Descripción detallada**

**1.1 Planteamiento del problema**

Los trastornos musculoesqueléticos (TME) constituyen una de las principales causas de discapacidad a nivel global, especialmente el dolor de espalda y cuello, los cuales generan una carga significativa en términos de productividad y costos para los sistemas de salud (OMS, 2023). Factores como la vida sedentaria, el uso intensivo de pantallas y el trabajo prolongado en posición estática han potenciado el desarrollo de posturas inadecuadas, entre ellas la postura cefálica adelantada (FHP), asociada a dolor cervical, disminución del control postural y fatiga muscular (Park et al., 2024; Kim et al., 2010).

El problema radica en que la mayoría de personas no son conscientes de su postura cotidiana y, por ende, no corrigen de forma oportuna los hábitos perjudiciales. Los correctores pasivos existentes (ej. fajas o soportes rígidos) resultan incómodos, poco estéticos y no contribuyen a la educación postural a largo plazo. Esto abre la oportunidad para desarrollar soluciones tecnológicas inteligentes basadas en sensores y algoritmos de inteligencia artificial que permitan monitorear, detectar y corregir la postura en tiempo real de manera no invasiva.

**1.2 Objetivo general**

Desarrollar un corrector inteligente de postura basado en hardware portátil, software y algoritmos de inteligencia artificial, que permita monitorear la postura corporal en tiempo real mediante conectividad Wi-Fi y brinde retroalimentación correctiva para la prevención de problemas musculoesqueléticos.

**1.3 Objetivos específicos**

1. Diseñar y construir un prototipo portátil equipado con sensores que capturen la posición y ángulos del tronco y la columna.
2. Implementar un sistema de comunicación inalámbrica mediante Wi-Fi que permita la conexión entre el dispositivo y una aplicación móvil.
3. Desarrollar una aplicación móvil multiplataforma para la visualización de datos, configuración del dispositivo y generación de reportes personalizados.
4. Entrenar e integrar algoritmos de inteligencia artificial que clasifiquen la postura como adecuada o inadecuada, y ajusten los umbrales de alerta según el usuario.
5. Validar la efectividad del sistema en un grupo de usuarios mediante pruebas piloto que evalúen su precisión, usabilidad e impacto en la corrección de hábitos posturales.

**1.4 Justificación**

La propuesta es relevante porque aborda una problemática de salud pública en aumento: los TME afectan tanto a trabajadores como a estudiantes, reduciendo productividad y bienestar (OMS, 2023). Desde una perspectiva académica y tecnológica, este proyecto es pertinente porque integra hardware portátil, software de gestión de datos y algoritmos de inteligencia artificial, enmarcándose en la tendencia global de soluciones en salud digital y tecnologías vestibles (Lee et al., 2023; Zhang et al., 2025).

Su impacto se proyecta en tres niveles:

* Salud: reducción de dolores musculoesqueléticos derivados de malas posturas.
* Educación: formación de hábitos saludables mediante retroalimentación en tiempo real y recordatorios digitales (Ma et al., 2024).
* Tecnología: innovación al combinar sensores inerciales, conectividad Wi-Fi y algoritmos de IA para personalizar la experiencia del usuario.

**1.5 Alcance**

El alcance inicial se centra en el diseño y validación de un prototipo funcional capaz de monitorear postura en tiempo real, emitir alertas correctivas y registrar datos en una aplicación móvil. No se busca en esta fase la producción industrial ni la comercialización masiva, sino la validación de viabilidad técnica y de usabilidad.

**2. Historias de usuario iniciales**

* **HU1:** Como usuario, quiero que el dispositivo me alerte cuando adopte una postura inadecuada, para poder corregirme de inmediato.
* **HU2:** Como usuario, quiero visualizar en mi aplicación móvil el historial de mis posturas, para conocer mi progreso en la corrección.
* **HU3:** Como usuario, quiero configurar los niveles de sensibilidad del dispositivo, para que se adapte a mis características físicas y necesidades.
* **HU4:** Como usuario, quiero recibir recordatorios de pausas activas cuando permanezca demasiado tiempo sentado, para reducir la fatiga muscular.
* **HU5:** Como administrador/investigador, quiero acceder a reportes agregados de varios usuarios, para evaluar la efectividad del sistema en la corrección postural.

**3. Especificaciones técnicas preliminares**

**3.1 Hardware**

* Microcontrolador ESP32 (conectividad Wi-Fi integrada).
* Sensores inerciales IMU (acelerómetro, giroscopio) MPU-6050
* Batería recargable de litio (autonomía mínima de 8 horas).
* Módulo vibratorio para alertas hápticas.
* Carcasa ligera y ergonómica (wearable, tipo banda o soporte dorsal).

**3.2 Software**

* Firmware en lenguaje C/C++ para adquisición y envío de datos desde el ESP32.
* Aplicación móvil multiplataforma (Flutter/Dark).
* Algoritmos de clasificación de postura basados en IA (modelo supervisado entrenado con datasets posturales).
* Conexión a base de datos en la nube (Firebase o AWS IoT).

**4. Diagrama de bloques inicial**

**Bloques principales:**

1. **Módulo de sensado:** IMU para capturar ángulos de columna y tronco.
2. **Módulo de procesamiento:** ESP32 encargado de procesar datos y comunicarse vía Wi-Fi.
3. **Módulo de retroalimentación:** vibrador háptico y notificaciones en la app.
4. **Aplicación móvil:** visualización de datos, configuración y reportes.
5. **Base de datos en la nube:** almacenamiento y análisis de registros históricos.

**5. Lista preliminar de componentes**

* 1 ESP32 DevKit.
* 1 IMU MPU-6050 o MPU-9250.
* 1 módulo vibrador mini.
* 1 batería recargable 3.7V Li-Po.
* 1 cargador de batería TP4056.
* Resistencias, cables y PCB.
* Material para carcasa (PLA impreso en 3D o similar).

1. **Diseño de encuesta de validación**

**Link encunta**

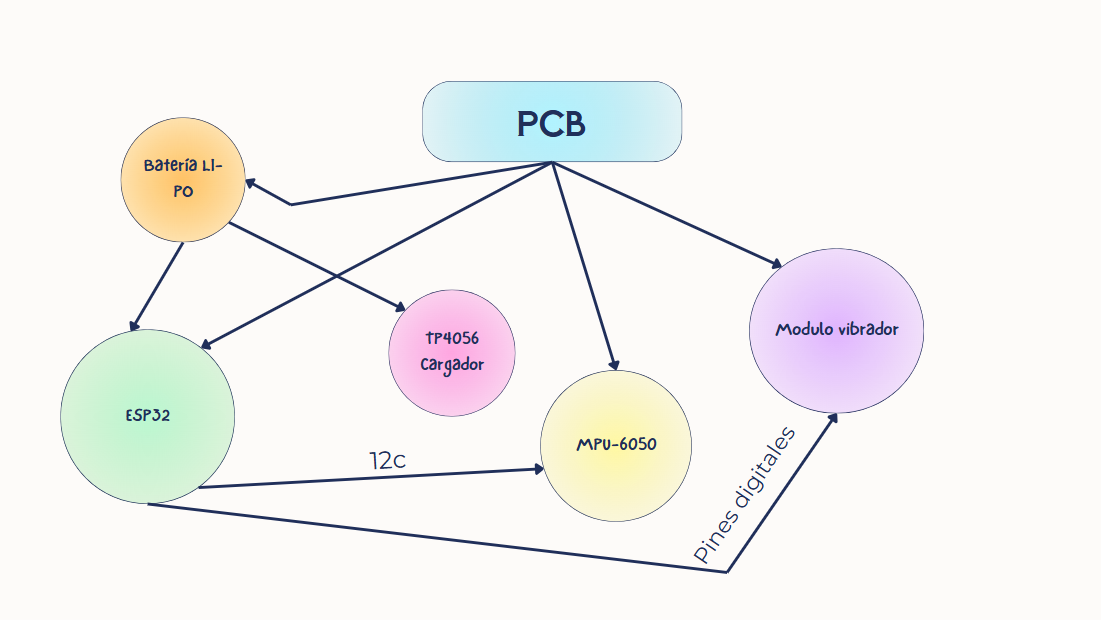
<https://docs.google.com/forms/d/e/1FAIpQLSf3__QuzKyStuC8AWRsMK30LuuDbkobWAltEoH8iIuBphB5yA/viewform?usp=header>

**Entrega 2 – Proyecto de Investigación Semana 2**

**Diseño esquemático del hardware.**

**Componentes Principales:**

1. Microcontrolador ESP32: Control central del dispositivo.
2. Sensor Inercial IMU (MPU-6050): Para capturar los ángulos de la columna y el tronco.
3. Módulo Vibrador: Para alertas hápticas.
4. Batería Li-Po: Fuente de energía.
5. Cargador de Batería TP4056: Para cargar la batería.
6. Conexiones: Resistencias, cables y PCB para interconexiones.



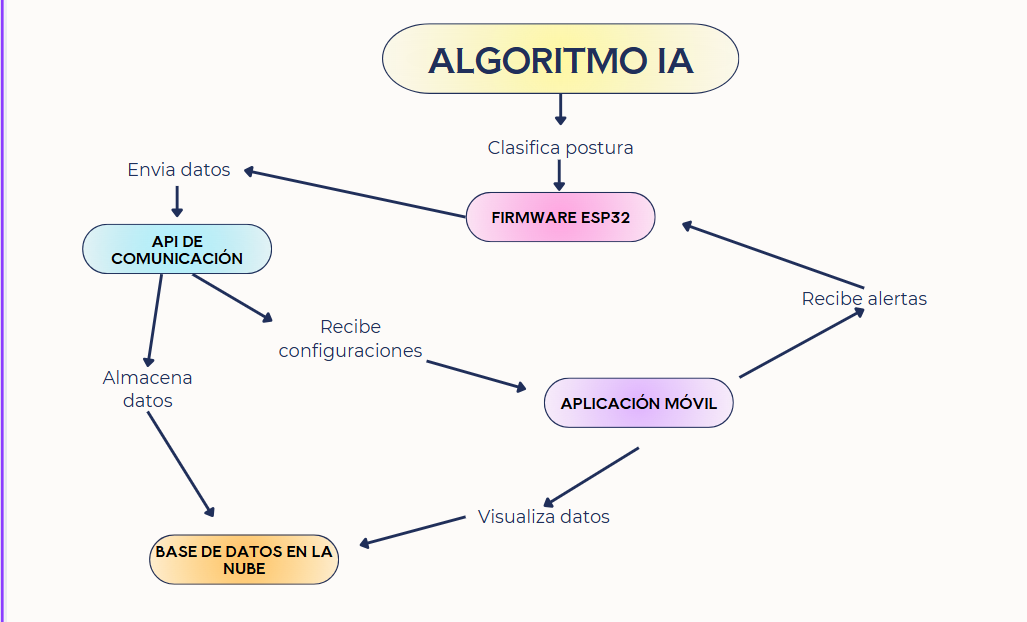
**descripción de Conexiones**

* **ESP32**: Actúa como el cerebro del dispositivo, procesando los datos de los sensores y enviando alertas.
* **MPU-6050**: Se conecta al ESP32 a través del protocolo I2C para enviar datos de movimiento.
* **Módulo Vibrador**: Se activa mediante los pines digitales del ESP32 cuando se detecta una postura inadecuada.
* **Batería Li-Po**: Alimenta el ESP32 y los otros componentes.
* **TP4056**: Permite cargar la batería de forma segura.
* **PCB**: Facilita las conexiones entre todos los componentes.

**- Diseño de arquitectura de software.**

**Componentes Principales**

1. Firmware del Dispositivo: Código que se ejecuta en el microcontrolador ESP32.
2. Aplicación Móvil: Interfaz de usuario para la visualización de datos y configuración del dispositivo.
3. Base de Datos en la Nube: Almacenamiento y análisis de datos históricos.
4. Algoritmos de Inteligencia Artificial: Procesamiento de datos para clasificar la postura.
5. API de Comunicación: Interfaz para la comunicación entre la aplicación móvil y la base de datos.



**Descripción de Componentes**

1. **Firmware del Dispositivo**:
   * Implementado en C/C++.
   * Captura datos del sensor (MPU-6050) y envía información a la API.
   * Recibe configuraciones y umbrales de alerta desde la aplicación móvil.
2. **Aplicación Móvil**:
   * Desarrollada en Flutter/Dart.
   * Permite a los usuarios ver su historial de posturas y recibir alertas.
   * Configura los niveles de sensibilidad y recibe notificaciones del firmware.
3. **Base de Datos en la Nube**:
   * Utiliza Firebase o AWS IoT.
   * Almacena datos históricos de posturas y configuraciones del usuario.
   * Proporciona acceso a la aplicación móvil para visualizar datos.
4. **Algoritmos de Inteligencia Artificial**:
   * Entrenados con datasets posturales.
   * Clasifican la postura como adecuada o inadecuada y ajustan los umbrales de alerta.
5. **API de Comunicación**:
   * Facilita la comunicación entre el firmware y la aplicación móvil.
   * Maneja solicitudes de datos y configuraciones.

**Ajuste de historias de usuario.**

1. **HU1: Alerta de Postura Inadecuada**
   * Como usuario,
   * Quiero que el dispositivo me alerte cuando adopte una postura inadecuada,
   * Para poder corregirme de inmediato y evitar problemas musculoesqueléticos.
2. **HU2: Visualización del Historial de Posturas**
   * Como usuario,
   * Quiero visualizar en mi aplicación móvil el historial de mis posturas,
   * Para conocer mi progreso en la corrección y mantener hábitos saludables.
3. **HU3: Configuración de Sensibilidad**
   * Como usuario,
   * Quiero configurar los niveles de sensibilidad del dispositivo,
   * Para que se adapte a mis características físicas y necesidades específicas.
4. **HU4: Recordatorios de Pausas Activas**
   * Como usuario,
   * Quiero recibir recordatorios de pausas activas cuando permanezca demasiado tiempo sentado,
   * Para reducir la fatiga muscular y fomentar un estilo de vida más activo.
5. **HU5: Acceso a Reportes Agregados**
   * Como administrador/investigador,
   * Quiero acceder a reportes agregados de varios usuarios,
   * Para evaluar la efectividad del sistema en la corrección postural y realizar mejoras basadas en datos.
6. **HU6: Personalización de Alertas**
   * Como usuario,
   * Quiero personalizar el tipo y la intensidad de las alertas (vibración, sonido, notificaciones),
   * Para que la retroalimentación sea más acorde a mis preferencias y no resulte molesta.
7. **HU7: Integración con Dispositivos de Salud**
   * Como usuario,
   * Quiero integrar el corrector de postura con otros dispositivos de salud (como pulseras de actividad),
   * Para obtener una visión más completa de mi bienestar físico.
8. **HU8: Entrenamiento de Algoritmos de IA**
   * Como investigador,
   * Quiero entrenar los algoritmos de IA con nuevos datasets posturales,
   * Para mejorar la precisión en la clasificación de posturas y adaptar el sistema a diferentes usuarios.

**Asignación de tareas.**

|  |  |  |
| --- | --- | --- |
| **Tarea** | **Responsable** | **Descripción** |
|  |  |  |
| Diseño del Prototipo | Jorge David Mercado Serje | Crear el diseño del prototipo del corrector de postura y seleccionar materiales. |
| Desarrollo del Firmware | Jorge David Mercado Serje | Programar el firmware en C/C++ para el ESP32, incluyendo la adquisición de datos. |
| Desarrollo de la Aplicación Móvil | Jorge David Mercado Serje | Crear la aplicación móvil en Flutter/Dart para la visualización y configuración. |
| Implementación de la Base de Datos | Mateo Gutiérrez Sepulveda | Configurar la base de datos en la nube (Firebase/AWS) para almacenar datos. |
| Entrenamiento de Algoritmos de IA | Mateo Gutiérrez Sepulveda | Desarrollar y entrenar los algoritmos de IA para la clasificación de posturas. |
| Pruebas de Usabilidad | Breyner Yulian Becerra Taba | Realizar pruebas de usabilidad y funcionalidad con usuarios reales. |
| Desarrollo de la Encuesta | Breyner Yulian Becerra Taba | Diseñar una encuesta para evaluar la efectividad del sistema y la satisfacción del usuario. |
| Análisis de Resultados de la Encuesta | Breyner Yulian Becerra Taba | Analizar los resultados de la encuesta y presentar hallazgos preliminares. |

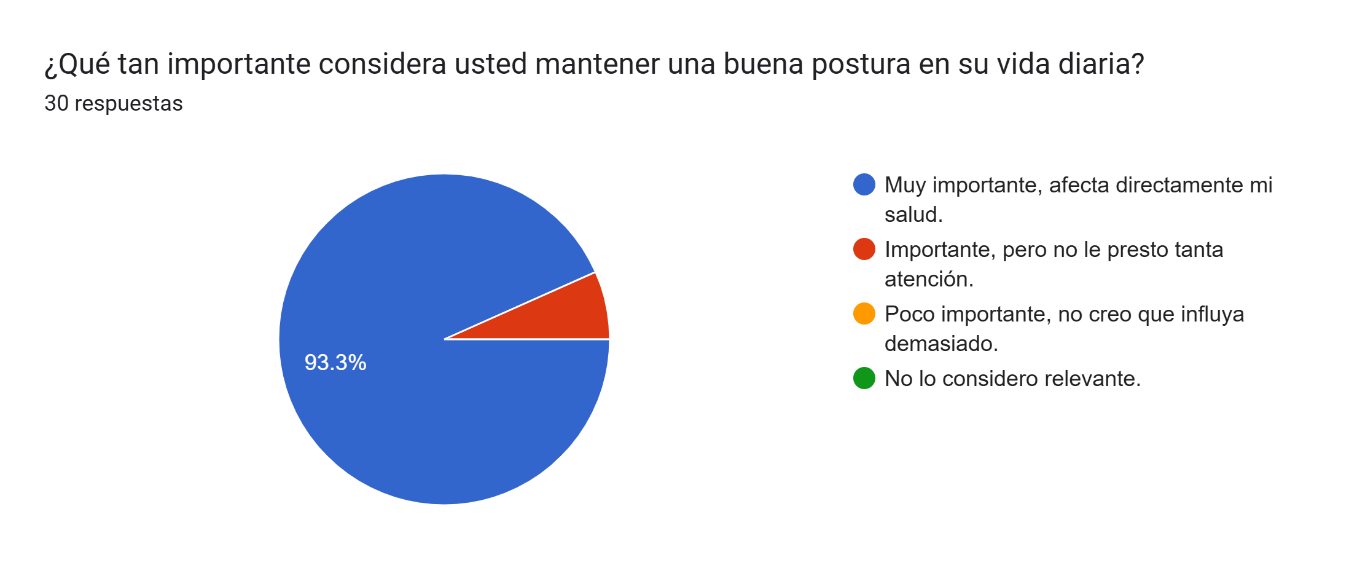
**- Lista definitiva de materiales.**

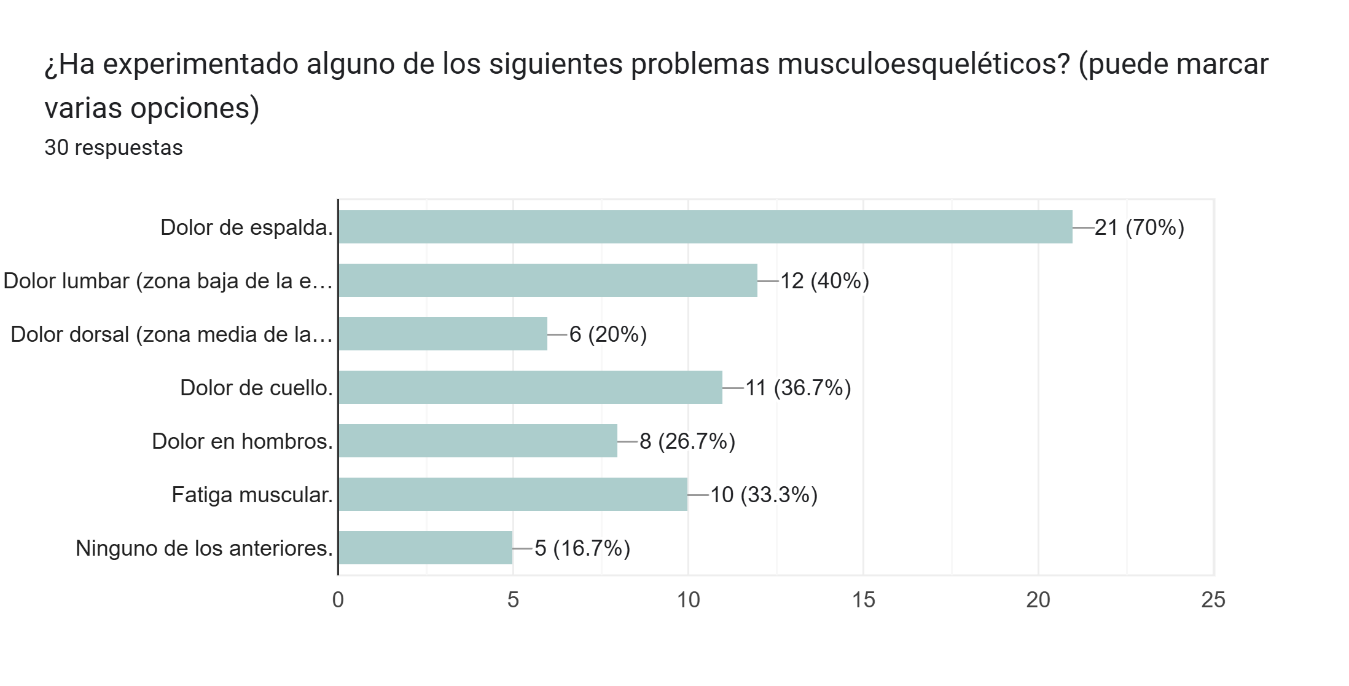
|  |  |  |
| --- | --- | --- |
| **COMPONENTES** | **CANTIDAD** | **DESCRIPCIÓN** |
| ESP32 DevKit | 1 | Microcontrolador con conectividad Wi-Fi integrada. |
| IMU MPU-6050 | 1 | Sensor inercial (acelerómetro y giroscopio) para capturar datos de postura. |
| Módulo Vibrador | 1 | Para alertas hápticas. |
| Batería Recargable Li-Po (3.7V) | 1 | Fuente de energía del dispositivo. |
| Cargador TP4056 | 1 | Cargador para la batería Li-Po. |
| Resistencias, Cables y PCB | Varias | Para conexiones y montaje del circuito. |
| Material para Carcasa (PLA impreso en 3D) | 5 | Para la fabricación de la carcasa del dispositivo. |
| Plataforma de Desarrollo (PC) | 1 | Para el desarrollo del firmware y la aplicación móvil. |
| Software de Desarrollo (IDE) | 1 | Herramientas necesarias para programar (ej. Arduino IDE, Flutter). |

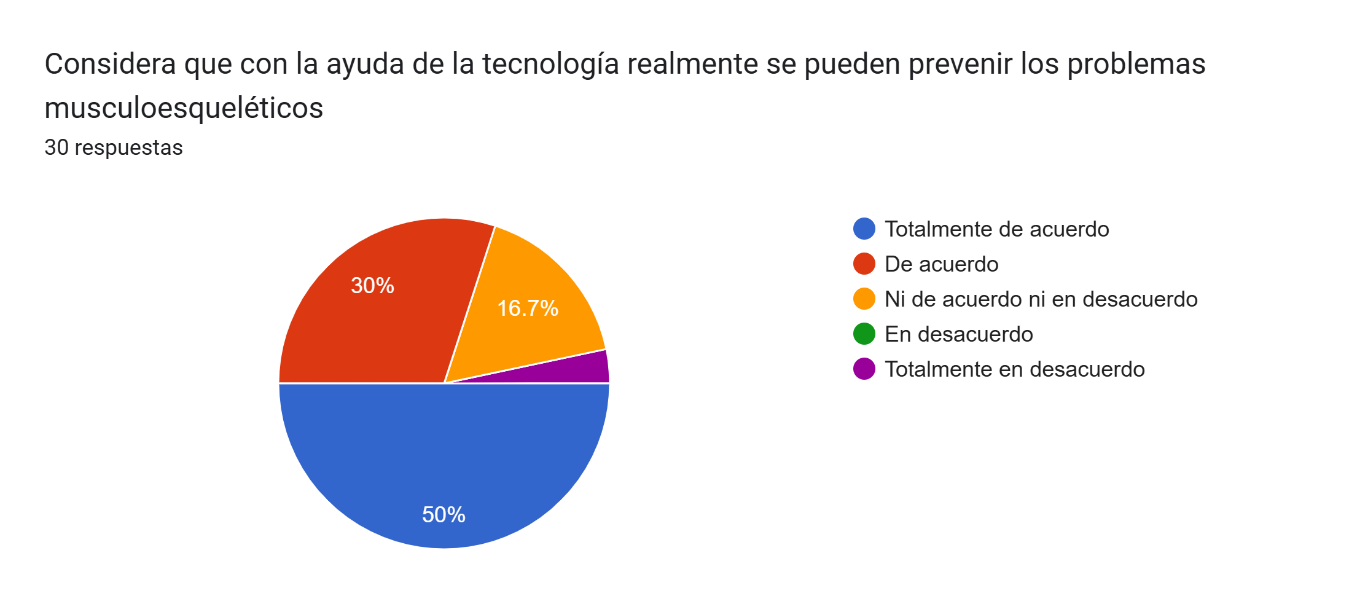
**- Aplicación de encuesta y análisis preliminar.**

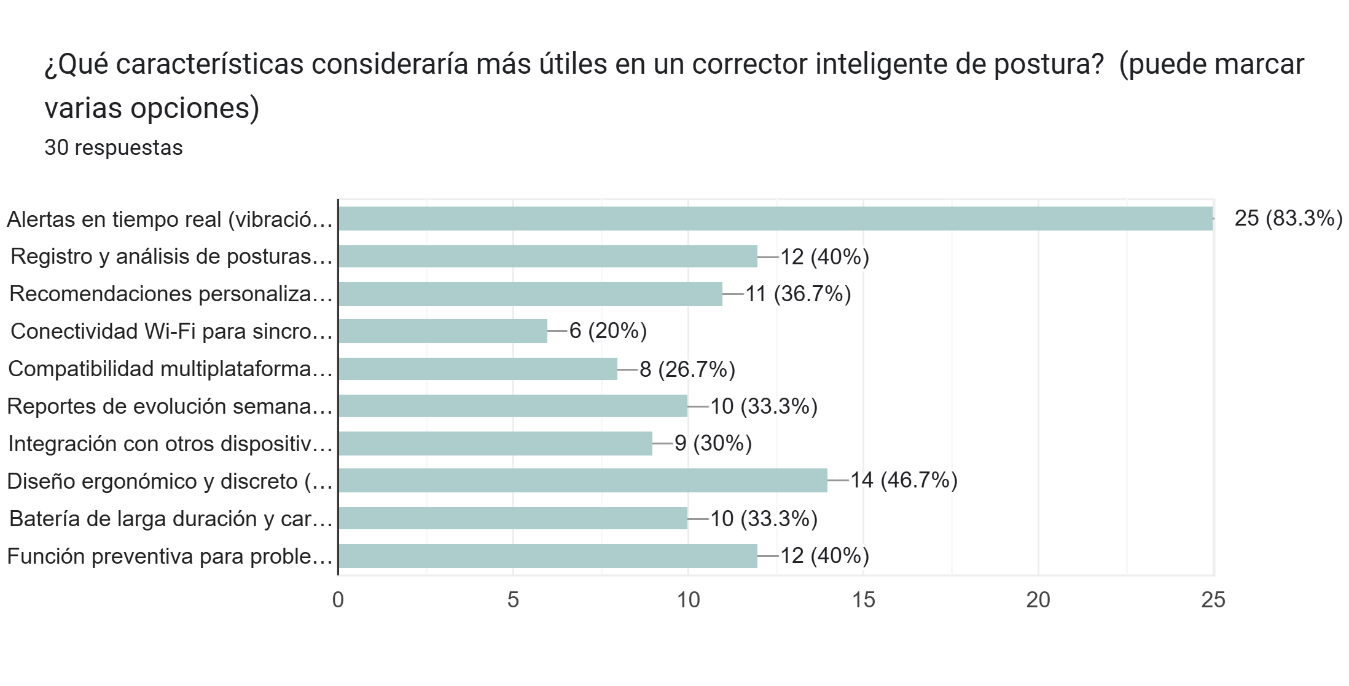
**Objetivo de la Encuesta**

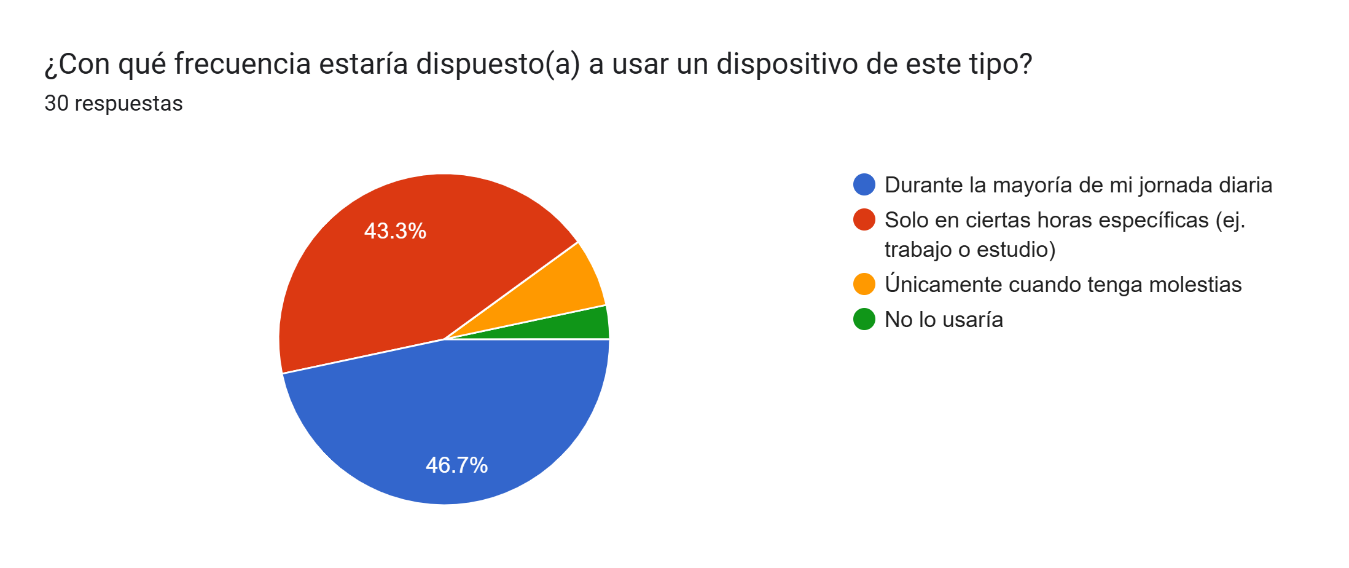
Recoger información sobre la experiencia del usuario con el corrector de postura, su efectividad y la satisfacción general.

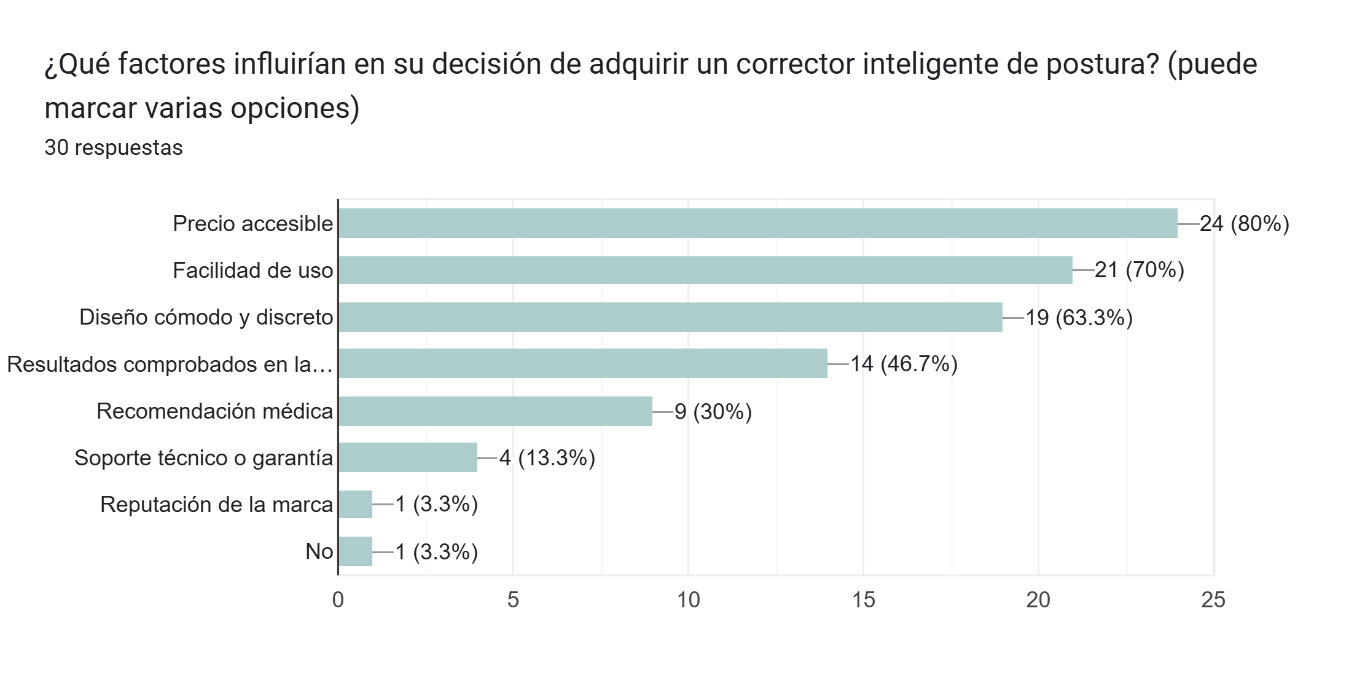












**Entrega 3 – Proyecto de Investigación Semana 3**

**Fichas técnicas de sensores y microcontroladores.**

Sensor: MPU-6050

* Tipo: Sensor Inercial (IMU)
* Funciones: Acelerómetro y giroscopio
* Rango de Medición:
  + Acelerómetro: ±2g, ±4g, ±8g, ±16g
  + Giroscopio: ±250, ±500, ±1000, ±2000 grados/segundo
* Precisión:
  + Acelerómetro: 16 bits
  + Giroscopio: 16 bits
* Interfaz: I2C
* Consumo de Energía: 5 mA (en modo activo)
* Costo Aproximado: $15.000 - $20.000

Microcontrolador: ESP32

* Tipo: Microcontrolador con Wi-Fi y Bluetooth
* Núcleos: Doble núcleo (Xtensa® 32-bit LX6)
* Frecuencia de Reloj: Hasta 240 MHz
* Memoria:
  + RAM: 520 KB
  + Flash: 4 MB (opcional)
* Conectividad: Wi-Fi 802.11 b/g/n, Bluetooth 4.2
* Consumo de Energía:
  + Modo activo: 160 mA
  + Modo de sueño profundo: 10 µA
* Costo Aproximado: $20.000 - $40.000

**Comparativa de opciones (precisión, costo, consumo, Wifi).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **COMPONENTE** | **OPCIÓN** | **PRECISIÓN** | **COSTO** | **CONSUMO** | **WI-FI** |
| Sensor | MPU-6050 | Alta (16 bits) | $15.000 - $20.000 | 5mA (activo) | NO |
| BNO055 | Muy Alta (16 bits) | $50.000 - $100.000 | 3.6mA (activo) | NO |
| LSM6DS3 | Alta (16 bits) | $30.000 - $50.000 | 1.5mA (activo) | NO |
| Microcontroladores | ESP32 | Alta | $20.000 - $40.000 | 160mA (activo) | SI |
|  | ESP8266 | Media | $10.000 - $30.000 | 160mA (activo) | SI |
|  | Arduino UNO | Baja | $20.000 - $60.000 | 50mA (activo) | NO |

**Justificación de selección.**

* Sensor MPU-6050:
  + Precisión: Ofrece una buena precisión para medir aceleraciones y giros, lo cual es esencial para detectar posturas inadecuadas.
  + Costo: Es una opción económica que se ajusta al presupuesto del proyecto.
  + Consumo de Energía: Aunque el consumo es moderado, es aceptable para un dispositivo portátil que no funcionará continuamente.
* Microcontrolador ESP32:
  + Conectividad: La capacidad de Wi-Fi y Bluetooth permite una fácil comunicación con la aplicación móvil y la nube.
  + Rendimiento: Su doble núcleo y alta frecuencia de reloj permiten un procesamiento rápido de datos, lo que es crucial para la clasificación en tiempo real de posturas.
  + Costo: Su costo es competitivo en comparación con otras opciones que ofrecen conectividad.

**Plan de pruebas iniciales.**

**Objetivo**

Validar el funcionamiento del corrector inteligente de postura y la efectividad de los sensores y microcontroladores seleccionados.

**Fases de Prueba**

1. Pruebas de Hardware:
   * Verificar Conexiones: Asegurar que todos los componentes estén correctamente conectados.
   * Medición de Consumo: Medir el consumo de energía en diferentes modos (activo, sueño profundo).
2. Pruebas de Sensores:
   * Calibración: Realizar la calibración inicial del MPU-6050.
   * Pruebas de Precisión: Comparar las lecturas del sensor con un dispositivo de referencia para validar la precisión.
3. Pruebas de Software:
   * Pruebas de Firmware: Cargar el firmware en el ESP32 y verificar la correcta adquisición de datos del sensor.
   * Pruebas de Comunicación: Probar la comunicación entre el microcontrolador y la aplicación móvil.
4. Pruebas de Usabilidad:
   * Pruebas con Usuarios: Realizar pruebas con usuarios reales para obtener retroalimentación sobre la efectividad del sistema y la interfaz de la aplicación móvil.
5. Análisis de Resultados:
   * Recopilar y analizar los datos obtenidos durante las pruebas para identificar áreas de mejora y realizar ajustes necesarios.

**Entrega 4 – Proyecto de Investigación Semana 4**

**Comparativa de frameworks y lenguajes.**

|  |  |  |  |
| --- | --- | --- | --- |
| **FRAMEWORK/LENGUAJE** | **DESCRIPCIÓN** | **VENTAJAS** | **DESVENTAJAS** |
| Flutter | Framework para desarrollo de aplicaciones móviles. | - Desarrollo multiplataforma. - UI atractiva y personalizable. - Buen rendimiento. | - Curva de aprendizaje si no se conoce Dart. |
| React Native | Framework para crear aplicaciones móviles usando JavaScript. | - Amplia comunidad. - Reutilización de código. - Integración con bibliotecas nativas. | - Dependencia de módulos nativos puede complicar el desarrollo. |
| Dart | Lenguaje de programación utilizado por Flutter. | - Sintaxis clara y concisa. - Buen rendimiento en aplicaciones móviles. | - Menor popularidad que JavaScript o Python. |
| JavaScript | Lenguaje de programación para desarrollo web y móvil. | - Muy popular y ampliamente utilizado. - Gran cantidad de bibliotecas y frameworks. | - Puede ser menos eficiente en aplicaciones intensivas. |
| C/C++ | Lenguajes de programación de bajo nivel. | - Alto rendimiento. - Control total sobre hardware. | - Mayor complejidad y tiempo de desarrollo. |
| Python | Lenguaje de alto nivel, fácil de aprender. | - Sintaxis simple. - Amplia biblioteca para IA y análisis de datos. | - Menor rendimiento en aplicaciones críticas de tiempo. |

**Definición de arquitectura de datos.**

La arquitectura de datos para el corrector inteligente de postura se puede definir en varias capas:

1. Capa de Adquisición de Datos

* Sensores: MPU-6050 para medir aceleración y giros.
* Microcontrolador: ESP32 para procesar datos y transmitirlos.

2. Capa de Procesamiento

* Firmware: Código en C/C++ que procesa los datos del sensor, detecta posturas y envía alertas.
* Algoritmos de IA: Clasificación de datos para identificar posturas adecuadas y inadecuadas.

3. Capa de Almacenamiento

* Base de Datos en la Nube: Firebase o AWS para almacenar datos históricos de posturas y configuraciones de usuario.

4. Capa de Presentación

* Aplicación Móvil: Interfaz en Flutter o React Native para que los usuarios visualicen su historial de posturas y reciban alertas.

**Plan de seguridad en transmisión WiFi.**

Para asegurar la transmisión de datos entre el microcontrolador (ESP32) y la aplicación móvil, se implementarán las siguientes medidas de seguridad:

1. **Autenticación**:
   * Uso de tokens de acceso para autenticar la comunicación entre el dispositivo y la aplicación.
2. **Cifrado**:
   * Implementar TLS/SSL para cifrar la comunicación entre el ESP32 y la base de datos en la nube.
   * Cifrado de datos sensibles almacenados en el dispositivo.
3. **Seguridad en la Red**:
   * Utilizar WPA3 para asegurar la red Wi-Fi utilizada por el dispositivo.
   * Asegurar que el dispositivo no se conecte a redes Wi-Fi no seguras.
4. **Actualizaciones de Seguridad**:
   * Implementar un mecanismo para actualizar el firmware del ESP32 y la aplicación móvil para corregir vulnerabilidades.
5. **Monitoreo**:
   * Implementar registros de acceso y monitoreo de la actividad para detectar intentos de acceso no autorizados.

**Borrador de API hardware-app.**

NOTA: FOTO DE LA APLICACIÓN

**Entrega 5 – Proyecto de Investigación Semana 5**

**Integración de investigación de hardware y software.**

La integración de hardware y software es fundamental para el correcto funcionamiento del corrector inteligente de postura. A continuación, se describen los componentes clave y su interacción:

**Componentes de Hardware**

* **Microcontrolador ESP32**: Actúa como el cerebro del dispositivo, procesando datos de los sensores y gestionando la comunicación.
* **Sensor MPU-6050**: Proporciona datos de aceleración y giroscopio para detectar la postura del usuario.
* **Módulo de comunicación (Wi-Fi)**: Permite la conectividad a la red para enviar datos a la aplicación móvil y recibir configuraciones.

**Componentes de Software**

* **Firmware del ESP32**: Desarrollado en C/C++, se encarga de la adquisición de datos, procesamiento y envío de información a través de Wi-Fi.
* **Aplicación Móvil**: Desarrollada con Flutter o React Native, proporciona una interfaz de usuario para visualizar datos de postura, configuraciones y alertas.
* **Base de Datos en la Nube**: Utilizada para almacenar datos históricos de posturas y configuraciones de usuario.

**Proceso de Integración**

1. **Desarrollo del Firmware**: Implementar el código que gestiona la adquisición de datos del MPU-6050 y la comunicación Wi-Fi.
2. **Desarrollo de la Aplicación Móvil**: Crear la interfaz que permita a los usuarios interactuar con el dispositivo, configurar alertas y visualizar el historial de posturas.
3. **Pruebas de Interoperabilidad**: Realizar pruebas para asegurar que el firmware y la aplicación móvil se comunican correctamente y que los datos se transmiten de manera efectiva.

**Propuesta de flujo de comunicación WiFi.**

El flujo de comunicación Wi-Fi entre el ESP32 y la aplicación móvil se puede describir de la siguiente manera:

**Flujo de Comunicación**

1. **Inicialización**:
   * El ESP32 se conecta a la red Wi-Fi utilizando credenciales predefinidas.
   * La aplicación móvil se conecta a la misma red Wi-Fi.
2. **Envío de Datos de Postura**:
   * El ESP32 adquiere datos del MPU-6050 y procesa la información de postura.
   * Se envía un POST a la API de la aplicación móvil con los datos de postura.
3. **Recepción de Configuraciones:**

* La aplicación móvil envía configuraciones (sensibilidad, alertas) al ESP32 mediante un POST a la API.
* El ESP32 recibe y aplica las configuraciones.

1. **Consulta del Historial:**

* La aplicación móvil solicita el historial de posturas enviando un GET a la API.
* El ESP32 responde con los datos almacenados en la base de datos en la nube.

1. **Alertas y Notificaciones:**

* Si se detecta una postura inadecuada, el ESP32 envía una notificación a la aplicación móvil a través de un sistema de alertas.

**Identificación de riesgos técnicos y plan de mitigación.**

A continuación, se presentan algunos riesgos técnicos asociados con el desarrollo del corrector inteligente de postura y sus respectivas estrategias de mitigación:

|  |  |  |
| --- | --- | --- |
| RIESGO TÉCNICO | DESCRIPCIÓN | PLAN DE MITIGACIÓN |
| Fallas en la Conexión Wi-Fi | Pérdida de conexión que impide la comunicación. | - Implementar reconexiones automáticas. - Monitorizar la calidad de la señal. |
| Inexactitud en los Datos del Sensor | Lecturas erróneas que afectan la detección de posturas. | - Calibrar el sensor regularmente. - Implementar filtros para suavizar los datos. |
| Problemas de Interoperabilidad | Incompatibilidad entre el firmware y la aplicación. | - Realizar pruebas exhaustivas de interoperabilidad. - Mantener documentación actualizada. |
| Vulnerabilidades de Seguridad | Riesgo de ataques cibernéticos en la transmisión de datos. | - Implementar cifrado TLS/SSL. - Actualizar regularmente el firmware para corregir vulnerabilidades. |
| Fallas de Hardware | Fallas en componentes físicos que afectan el funcionamiento. | - Realizar pruebas de estrés en el hardware. - Tener un plan de respaldo para reemplazo de componentes. |
| Problemas de Usabilidad en la Aplicación | Dificultades para los usuarios al interactuar con la app. | - Realizar pruebas de usabilidad con usuarios reales. - Iterar en el diseño según la retroalimentación. |

**Entrega 6 – Proyecto de Investigación Semana 6**

**Diagramas UML (casos de uso, clases, secuencia).**

**Modelo de base de datos.**

**Plan de integración hardware–software vía WiFi.**